```
pragma solidity ^0.5.3;

contract Operators{


    address public owner;
    bool public flag;
    constructor() public{
        owner = msg.sender;
    }

    // ==
    modifier OwnerOnly{
        if (msg.sender == owner){
            _;
        }
    }
    //< , >

    function ifElse(uint x, uint y) public pure returns(uint result){

        if (x > y){
            result = x - y;
            return result;
        }else if (x < y){
            result = y - x;
            return result;
        }else{
            return 0;
        }
    }

    // !=

    function NotEquals( int x, int y) public pure returns(bool){
        if (x != y){
            return true;
        }
    }

    //>=

    function GE( int x, int y) public pure returns(bool){
        if (x >= y){
            return true;
        }
    }

    //<=
    function LE( int x, int y) public pure returns(bool){
        if (x <= y){
            return true;
        }
    }

    //AND &&
    function AndOperator( int x) public pure returns(bool){
        if (x >5 && x < 10){
            return true;
        }
    }

    //OR ||
    function OrOperator( int x, int y ) public pure returns(bool){
        if (x >5 || y > 10){
            return true;
        }
    }

    //Operator !
    function Not(int x) public returns(bool){
        if (x > 5){
            flag = true;
            return flag;
        }

        if(!flag){
            return flag;
        }
    }

    // + , - / , %, *

    function MathsOp(int x, int y, int z) public pure returns(int 
result){
//         result = (x + y) * z; //( 3+ 5) * 6 = 8 * 6 = 48
//         result = x + y * z; // 3+ 5 * 6 = 3 + 30 = 33
//           result = (x + y) / z; // (3+10) / 6 = 13 / 6 = 2
           result = (x + y) % z; // (3+10) / 6 = 13 / 6 = 1
//           result = x % y;
//           return result;
    }

}
```